Flexible Server Selection in Widely Distributed Environments

Shimokawa, Toshihiko
Department of Computer Science and Communication Engineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

Yoshida, Norihiko
Department of Computer and Information Sciences,Nagasaki University

Ushijima, Kazuo
Department of Computer Science and CommunicationEngineering, Graduate School of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1500430

出版情報：九州大学大学院システム情報科学紀要. 5 (1), pp.7-12, 2000-03-24. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第5巻 第1号 平成12年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.5, No.1, March 2000

Flexible Server Selection in Widely Distributed Environments*

Toshihiko SHIMOKAWA**, Norihiko YOSHIDA***and Kazuo USHIJIMA**

(Received December 10, 1999)

Abstract: Many service providers prepare multiple servers to cope with over loading. In such cases, the problem is how to select the best server out of them. Existing server selection models impose user extra effort or have less flexibility. In this paper, we describe a new server selection mechanism. We add server selection function to DNS. This mechanism is flexible, and has wide applicability.

Keywords: Internet, Load balancing, DNS, Name Server, Tenbin

1. Introduction

There are very many services on the Internet these days. As the number of users has been increasing dramatically, we face to a problem of server overloading. To solve this, service providers often prepare multiple servers. Now the problem is how to distribute requests among these servers. Put another way, the problem is how to select best server out of them.

Server selection mechanisms used currently impose heavy tasks on users, or have less flexibility. Consequently, we study on flexible server selection mechanism for effective load balancing among multiple servers. To realize this, we introduce "query preprocessor" attached to a Domain Name System(DNS) server. It can coordinate various server evaluation methods and selection mechanisms. It works as a proxy server, and imposes no need of modifying working DNS network.

The rest of this paper is organized as follows. Section 2 overviews the server selection models which are already used or proposed. We show our flexible server selection mechanism in Section 3. We evaluate our mechanism in Section 4. Section 5 overviews related works. Section 6 is a conclusion.

2. Server selection models

2.1 The ideal model

There are many kinds of services that suffer server overload and require server multiplication. Therefore, the model should not depend on the kind of service.

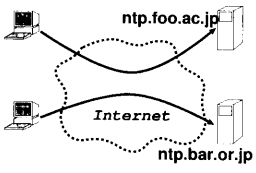

Fig.1 Client side selection model

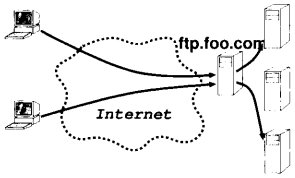

Fig.2 Server side selection model

Multiplied servers may locate in wide area. Therefore, the model should be able to implement distributedly.

The model should be scalable, as the server selection should not be a new bottleneck.

2.2 Existing models

We classify server selection models by where to select a server.

1. Client side selection model
2. Server side selection model
3. Intermediate system selection model

2.3 Client side selection model

In this model, a client, or a user of the client, selects a server(**Fig.1**). The client or the user is assumed to have enough knowledge to distribute the load among the multiplied servers. This assumption is unrealistic. The user may be imposed an extra effort.

2.4 Server side selection model

In this model, a server that receives a request makes a selection(**Fig.2**). This front-end server, which we name the "main server", forwards the request to the best server out of back-end multiplied servers.

This model is flexible because it can adapt various

* This research was supported by the JSPS-RFTF-96P00603.

** Department of Computer Science and Communication Engineering

*** Department of Computer and Information Sciences, Nagasaki University

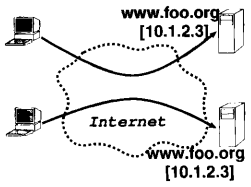

Fig.3 Network selection model

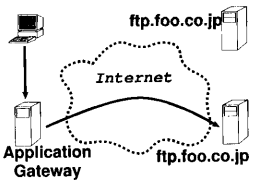

Fig.4 Application gateway selection model

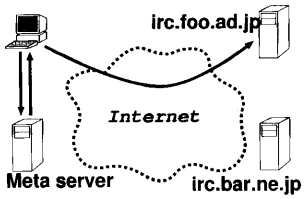

Fig.5 Meta server selection model

situations. However, the main server may become a new bottleneck, therefore this model lacks scalability.

2.5 Intermediate system selection model

There can be many intermediate systems between clients and servers. Such intermediate systems select a server in this model. We classify this model into the below three sub-models according to network layers where selection takes place.

2.5.1 Network selection model

In this model, the network itself selects a server(**Fig.3**). Anycast in IP version 6 is one of the implementations of this model.

All network-based applications use the network. Therefore, this model has the best availability. However, technologies to implement this model are not enough matured. This model lacks flexibility of server selection mechanisms.

2.5.2 Application gateway selection model

There are many intermediate systems between clients and servers in the application layer. Proxy servers at firewalls or cache servers to reduce communication traffics are typical examples of these intermediate systems. We call these application-layer intermediate systems "application gateways' .

In this model, an application gateway selects a server(**Fig.4**). Flexibility of this model is the same as flexibility of the application gateway. For employing this model, all applications must use the application gateway, therefore this model lacks applicability.

2.5.3 Meta server selection model

There are meta services used before real service such as DNS, SLP(Service Location Protocol), URL Resolver[1], and so on. In this model, servers for such meta services select a server. Flexibility and applicability of this model depend on flexibility and applicability of the meta services.

2.6 Widely used models

The client side selection model and meta server selection model using round robin DNS are two widely used models today.

We show weak points of the client side model in Section 2.3. This model is easy to implement, however not appropriate.

The meta server selection model using round robin DNS has wide applicability. Almost all applications use DNS to resolve hostnames to IP addresses. Therefore, this approach has wide applicability. It is transparent and independent from the kind of service, client software and server software. DNS servers work distributedly. Therefore, they do not become a bottleneck. However this model has less flexiblity. This model can select a server using a simple round robin algorithms only.

3. Flexible server selection using DNS

We propose new server selection mechanism[2]. We add flexible server selection capability to DNS. Almost all applications use DNS to resolve hostnames to IP addresses. Therefore, this approach has wide applicability. It is service, client software and server software independent and transparent. DNS servers work distributedly. Therefore, they do not become a bottleneck.

We introduce "query preprocessor" attached to existing DNS network. Our "query preprocessor" is desinged as a proxy server. We show a detail at Section 3.1.

Many flexible server evaluation criterions can integrate into "query preprocessor." Therefore, our method can adopt many situations. We show these criterions at Section 3.2.

"Query preprocessor" can act server selection at both server side and client side. Many of other server selection methods can act on server side only. We show a detail in next section. Therefore, if service providers prepare multiple servers and they do not prepare any sophisticated server selection mechanisms, users cannot enjoy these multiple servers enough. However, at this situation, users can use

"query preprocessor" at client side. Then, they enjoy multiple servers.

When selecting on client side, the problem is how to make candidates for selection. Because to know which hosts are multiplied servers is not obvious. We show the solution at Section 3.3.

3.1 Separate preprocessor

Now we introduce a "query preprocessor". An existing DNS server contains both hostname database and query processor. When it receives a request, it searches hostname database and returns an answer.

We separate these two tasks. "Query preprocessor" not only answers to requests, but also selects an IP address for the hostname using some server selection criterions. It works as a proxy server, and imposes no need of modifying working DNS network.

Clients must change only one configuration item to use "query preprocessor." The item is a DNS server they use. If clients use DHCP or PPP, the DNS server should be configured automatically. Therefore, in these cases, users do not need be aware of this configuration change.

3.2 Criterions

There are various criterions for selecting a server: for example, round trip time, through put, load of servers, and so on. In this paper we focus on network status like round trip time. Because, we can evaluate it easily from client side.

"Query preprocessor" communicates with "observer" to collect network status. The observer probes network status periodically. There are two observation methods. One is active observation, and the other is passive observation.

3.2.1 Active observation

In this method, the observer collect network status actively, for example:

- observe by probe packets
- observe by service packets

The service packets mean packets which client and server communicate. The observer sends some probe packets(ICMP echo request, for example) and measures their round trip time. It uses the results to evaluate network status.

If a service that a client wants to use is known beforehand, the observer can use this real service to probe. For example, the observer may request WWW servers to transfer some contents and measure its transfer time. However, there are no general methods to know the service beforehand. Therefore, we assume a service based on a hostname alias defined in RFC2219[3]. For example, if a client requests resolving a hostname that begins with "WWW", then "query preprocessor" assumes that the client wants to use WWW(http)service. Therefore, the observer can try to use http packets to evaluate network.

One of weak point of the active observation method is that observer generates useless traffics. If we use service packets, traffic and server load is not negligible. On the other hand, probe packets like ICMP echo are often filtered out by firewalls; probe packets cannot reach servers, and the observer cannot evaluate network, while service packets are rarely filtered out, and the observer can evaluate network.

3.2.2 Passive observation

In this method, the observer passively observes network status. Merit of this method is small network overhead. Examples of passive observation targets are listed below.

- traffic of service packets
- routing information

When observing these traffic of service packets, network evaluation would be more accurate.

Routing information contains information to select a network route. The observer can use this information to evaluate network. For example, BGP packets contain AS path list. The observer can compare all AS path length for all of multiple servers.

3.3 Candidates

When running "query preprocessor" on server side, getting candidates is easy work. However running on client side, it is not easy. We have to collect candidates by some way. Actual candidates are IP addresses of the corresponding servers.

We collect candidates from DNS in the first place. There are four cases how multiplied servers' hostnames and IP addresses stored in DNS.

1. Single hostname and single IP address
2. Single hostname and multiple IP addresses
3. Multiple hostnames and single IP address
4. Multiple hostnames and multiple IP addresses

In the case 1, we can collect only one IP address and one hostname. The only candidate is this IP address. Therefore, we cannot act selecting.

In the case 2, we can collect all the multiple IP addresses from DNS. We use these as candidates.

In the case 3, we can collect only one IP address like case 1. However, it is unusual to use this for load balancing. Frequently, it used for virtual host-

ing.

In the case 4, we want to collect all the multiple IP addresses. To do this, we have to collect all the hostnames at first. However, unfortunately, there are no general methods to do it. Collecting these hostnames automatically is impossible. Therefore, we collect these by hand. We show a detail in next section.

3.3.1 Multiple hostnames and multiple IP addresses

It is common to prepare multiple servers for load balancing. Giving them indivisual hostnames is not special, in these days. There are no general methods to know these hostnames automatically. We must list up these hostnames by hand. Then, resolve these hostnames to IP addresses. We use these IP addresses for candidates.

We use all these IP addresses for candidates to resolve all these hostnames. "Query preprocessor" may answer an IP address that a requested hostname does not have.

Suppose for example that there are two IRC servers "irc.foo.org" and "irc.bar.net." The IP address of irc.foo.org is 133.5.0.1, and irc.bar.net is 192.50.13.250. When "query preprocessor" receives request to resolve irc.foo.org, it use either 133.5.0.1 or 192.50.13.250 for candidates. It can answer 192.50.13.250 that irc.foo.org does not have.

"Query preprocessor" can cope with multiple hostnames and multiple IP addresses. However, there is a limitation of applicability because we have to make a list by hand.

4. Consideration

4.1 Implementation of prototype system

We implement a prototype system. We call this "Tenbin[†1]". Tenbin contains a request receiver, a criterion database, a criterion decider, and some server evaluators, each of which has a selection criterion.

When the request receiver receives a request from clients to resolve a hostname, Tenbin parses the request. If Tenbin does not know type of request, Tenbin forwards the request to predefined existing DNS server. The criterion decider selects a server-selecting criterion for other requests. The decider searches the criterion database using the requested hostname. In current implementation, we must register all the hostnames that they want to use Tenbin's server-selecting mechanism beforehand.

All the server evaluators have their own evaluation executors. Each evaluation executor communicates with the observer to obtain some information they need, if necessary. Tenbin passes the request to a decided evaluation executor, and it evaluates servers and resolves the hostname to IP address.

When there is no server evaluator for the request, Tenbin uses a default server evaluator. The current default server evaluator simply forwards the received request to predefined existing DNS server.

This is how Tenbin works. The current implementation uses the active observation method using probe packets, and the observer is embedded in the server evaluator. We use an object-oriented script language Ruby[4] for coding Tenbin.

†1 Tenbin stands for "Tenbin is Experimental Name server for load Balanced Internet"

Table 1 Selection counts at Kyushu Univ.

hostname	count	ratio(%)
ring.nacsis.ac.jp.	1486	51.1
ring.ocn.ad.jp.	602	20.7
ring.etl.go.jp.	223	7.68
ring.aist.go.jp.	150	5.16
ring.asahi-net.or.jp.	130	4.47
ring.so-net.ne.jp.	124	4.27
ring.crl.go.jp.	74	2.54
ring.jah.ne.jp.	55	1.89
ring.shibaura-it.ac.jp.	43	1.48
ring.omp.ad.jp.	9	0.31
ring.ip-kyoto.ad.jp.	9	0.31
ring.htcn.ne.jp	1	0.03

4.2 Evaluation using prototype system

We examined Tenbin's capabilities and performance on some networks.

4.2.1 Result of selected hosts

We examined transition of server selection results. In this examination, we used ftp.ring.gr.jp.[5] for server selection targets. The ftp.ring.gr.jp has 14 IP addresses, and they locate at 14 different networks. We measured round trip time to each of them every 30 minutes by sendig ICMP ECHO packets. We show the result in **Table 1**, and plot selected servers in **Fig.6**.

The graph in **Fig.6** illustrates that round trip time between Tenbin and target servers does not change often. Therefore, selected server does not change often usually.

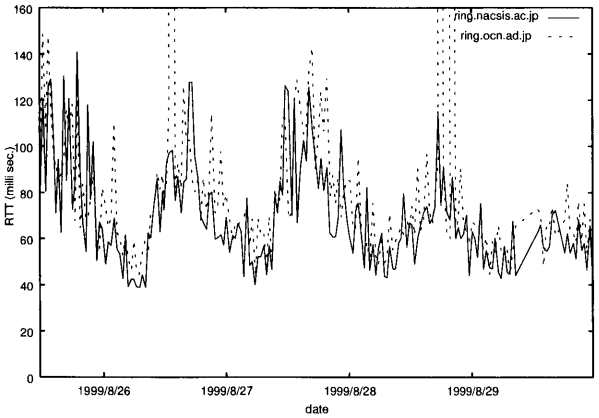

Fig.6 round trip times

4.2.2 Result difference by location

Next, we studied difference of selected host by location of Tenbin. We locate Tenbin some locations. We show the result of selected hosts at Nagasaki University(**Table 2**), WIDE Project FUKUOKA NOC(**Table 3**) and WIDE Project NEZU NOC(**Table 4**). These tables show only top three hosts of selected hosts.

We see from **Table 1** and **Table 2** that these two results are similar. From a network topology standpoint of view, Kyushu University and Nagasaki University locate at same AS(Autonomous System).

We also see from these two tables that a tendency of selected hosts show concentration.

From a geographical location, Kyushu University and WIDE Project FUKUOKA NOC are locate in the same campus. However from a network topology standpoint of view, they are locate at different ASs. We see from **Table 1** and **Table 3** that these two results are very different.

From a view point of network topology, WIDE Project FUKUOKA NOC and WIDE Project NEZU NOC locate at same AS. However these two result show different tendency. We think this difference comes with difference of connectivity between other ASs.

We see from **Table 3** that the tendency of selected hosts does not show concentration. This comes from a change of network topology.

For example, topology between WIDE Backbone and NSPIXP3 †2 changed on October 14. **Table 5** is a result of selected hosts before October 14. It shows a tendency.

These results show that Tenbin selects the nearest server automatically. Users can use the best server automatically.

†2 one of major Internet exchange in Japan

Table 2 Selected times at Nagasaki Univ.

hostname	ratio(%)
ring.nacsis.ac.jp.	88.2
ring.ocn.ad.jp.	10.0
ring.shibaura-it.ac.jp.	0.53

Table 3 Selected times at WIDE FUKUOKA NOC

hostname	ratio(%)
ring.shibaura-it.ac.jp.	20.0
ring.asahi-net.or.jp.	16.4
ring.omp.ad.jp.	10.6
ring.ocn.ad.jp.	10.6

Table 4 Selected times at WIDE NEZU NOC

hostname	ratio(%)
ring.asahi-net.or.jp.	58.0
ring.nacsis.ac.jp.	29.1
ring.ocn.ad.jp.	12.2

Table 5 Selected times at WIDE FUKUOKA NOC (before 1999/10/14 10:00)

hostname	ratio(%)
ring.omp.ad.jp.	58.3
ring.ip-kyoto.ad.jp.	22.1
ring.htcn.ne.jp.	11.0

4.2.3 Overheads

We examined an overhead of Tenbin. When Tenbin forwards a request to the existing DNS server, average overhead time of Tenbin was 9.1 ms. We think this overhead is negligible.

4.3 Limitation

4.3.1 Specific hosts

There can be a case that users or administrators want to use a specific server out of multiple servers. Our DNS server may select another host. There are two solutions for this problem.

1. Use an IP address directly instead of the hostname. It is difficult to use the IP address for ordinary users. However, the case which users want to use a specific server must be a special case, administrative work for example. Therefore, this is practical.
2. Giving individual hostnames one by one. Users can access a specific host using its specific hostname. However, if users use this specific

host to receive ordinary service, load balancing does not function.

4.3.2 Select using only hostname

To use a service, information other than hostname is required: the port number of TCP/UDP, path name of contents for example. If these are different between multiplied servers, our proposal method is not usable.

For instance, some of IRC servers provide services on multiple port for large number of clients. However, not all servers provide such service. Therefore, we cannot make all these IRC servers as candidates.

5. Related works

Smart Clients[6] is a one of client side selection models. Smart Clients use Java applet to deliver server status to clients. This system have to modify both server and client softwares. It lacks applicability.

SWEB[7] is a mixture of round-robin DNS and server selection model. In this system, server selection was done by two steps. First step, DNS server selects a first server by round-robin. Second step, the selected first server selects a final server out of multiple servers and re-direct request to the final server. This system can evaluate server status on the first server. However, to evaluate server status, servers has to modify to collect server status. Therefore, it lacks applicabity.

The system by Shigechika et al. [8] uses network selection model. It assigns a single IP address to multiple servers, and lets routing mechanism select an appropriate server.

The weighted round robin algorithm[9] is a simple extension of round-robin DNS. It assigns multiple IP addresses to a server according to servers' capacity. It can evaluate servers' capacity and its load occasionally. However, it cannot evaluate network capacity and load.

Approach of Cluster DNS[10] is similar to ours. It add server selection mechanism to DNS. However it, and all of above except round-robin DNS, assume that they, at least a part of them, run on server side. Therefore, if service providers do not use these systems, users cannot enjoy multiple servers.

Distributed Director[11] is a product for load balancing. It uses the applicabity gateway selection model and the meta server selection model. It use routing information to select a server. If service providers cannot get enough routing information, they cannot perform selection.

6. Conclusion

In this paper we propose a design for flexiblely server selection mechanism using DNS server that embeds various server evaluation criterions. This system design has wide applicability, and runs distributedly. We implement a prototype system, and evaluate it.

The future direction of this study will include:

- Detailed result analysis
- Estimate using widely distributed service
- Design and implement various criterion
- Develop better criterion selection mechanisms

Current Tenbin implements only active observation with probe packets. We must implement other observation methods, and evaluate these.

Acknowledgments

We would like to thanks Mr. Hiroshi Esaki(Tokyo University), and Mr. Ikuo Nakagawa (INTEC Systems Laboratory) for offering host for running Tenbin at WIDE Project NEZU NOC.

References

1) S. Kurihara, T. Hirotsu, T. Takada, S. Sugawara, "Adaptive routing selection for the URL resolver, " The Seventh Workshop on Multiagent and Cooperative Computation, Dec. 1998 (in Japanese)
2) T. Shimokawa, N. Yoshida, K. Ushijima, "Flexible Load Balancing Mechanism using Name Server," Internet Conference '99, Dec. 1999 (in Japanese)
3) M. Hamilton, R. Wright, "Use of DNS Aliases for Network Services," RFC 2219, Oct. 1997
4) Y. Matsumoto, "Ruby the Object-Oriented Script Language, " http://www.ruby-lang.org/
5) Ring Server Project, http://www.ring.gr.jp/
6) C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, D. Culler. *Using Smart Clients to Build Scalable Services.*, Usenix '97.
7) D. Andresen, T. Yang, V. Holmedahl and O. Ibarra., *SWEB: Towards a Scalable WWW Server on Multi-Computers*, Proceedings of the 10th International Parallel Processing Symposium, April, 1996.
8) N. Shigechika, O. Nakamura, N. Sasakawa, J. Murai, "Network and Information Providing System for Nagano Olympic" Journal of Information Processing Society of Japan Vol.39 No.2, 1998(in Japanese)
9) T. Baba, S. Yamaguchi, "A DNS based implementation on widely load balancing mechanism, " IPSJ SIG Notes, 98-DSM-9, pp.37-42, May 1998 (in Japanese)
10) V. Cardellini, M. Colajanni, P.S. Yu, *DNS dispatching algorithms with state estimators for scalable Web-server clusters*, World Wide Web Journal, Baltzer Science, vol. 2, no. 3, July 1999, pp. 101-113.
11) CISCO Systems Inc. DistributedDirector, http://www.cisco.com/warp/public/cc/cisco/mkt/scale/distr